A dynamic programming algorithm for the ATM network installation problem on a tree

A Dynamic Programming Algorithm for the ATM Network Installation Problem on a Tree

Robert L.M.J. van de Leensel[1,2] Olaf E. Flippo [1] Arie M.C.A. Koster [1]

February 10, 1998

Abstract

This paper considers the ATM Network Installation Problem on a tree. To install such a communication network, decisions concerning the location of hardware devices, the capacity installation on links, and the routing of demands have to be made simultaneously. The problem is shown to be NP-hard. By exploiting the tree structure we show that the problem can be solved to optimality using a pseudo-polynomial time dynamic programming algorithm. Computational experiments on real-life problem instances indicate that the algorithm is highly efficient.

1 Introduction

Modern telecommunication networks are capable of processing multiple telecommunication services on a single physical network. These so-called broadband networks usually consist of several hierarchical network layers. At the top layer (often referred to as the Backbone), large capacity nodes serve large geographical areas. These areas are decomposed into smaller regions, each of which is served by nodes located in lower layers of the network. In the Backbone, connectivity between nodes is usually fairly high. This is due both to high traffic requirements, which make the installation of direct links between nodes economically attractive, and to reliability considerations, which coerce that in case of a calamity in the network (the breakdown of a node or a link), alternative routing between pairs of nodes must be available. In lower levels of the network both the intensity of traffic and the need for reliability decrease, and the structure commonly used here is thus a tree. For a detailed description of telecommunication networks, as well as an explanatory visualization, see Balakrishnan et al. [1].

In order to send information of different types of services over the same broadband network, a structured protocol is required. Nowadays ATM (Asynchronous Transfer Mode) is widely

[1]Dept of Quantitative Economics, Maastricht University, P.O.Box 616, 6200 MD Maastricht, The Netherlands.
[2]e-mail: r.vandeleensel@ke.unimaas.nl

accepted as the standard protocol for (future) broadband networks. In the ATM protocol, a message that needs to be sent from its source to its destination is decomposed into small units of information which are stored in ATM cells. In addition to a part of the message, an ATM cell also contains certain routing information, which enables the cell to be routed through the network from source to destination. At the destination node, the different units are combined to reconstruct the original message. To implement the ATM protocol on the telecommunication network, certain hardware devices have to be installed, connections between these devices have to be made, and customers must be connected to these devices. As such, an ATM network topology is built upon an existing network structure. The problem of designing and maintaining telecommunication networks typically involves simultaneous decisions regarding the location of hardware devices, the capacity of these devices, the capacity installation on connections between devices, and the routing of messages over these connections. Since these problems are by far too complex to be handled at once, they are often decomposed.

One way to decompose network problems is to consider location, capacity installation and routing problems separately. Gavish [10] schematizes the decomposition of the overall network design process and gives references to literature on the individual problems. More recently, combined routing and capacity installation problems have been studied, although most studies are restricted to the Backbone network. Gavish and Altinkemer [11] consider a nonlinear model incorporating queueing costs as well as fixed and variable transportation costs on connections, and use a Lagrangean decomposition approach to tackle the problem. Brockmüller, Günlük and Wolsey [5] use an integer linear formulation and apply polyhedral techniques in a branch-and-cut framework. Both models assume that each commodity (i.e. a demand which must be sent from its source node to its destination node) should be routed via a single path (so called non-bifurcated routing). In constrast, Magnanti, Mirchandani and Vachani [13],[14] allow for the demand of a commodity to be split among different paths, thereby generating flow models (bifurcated routing). In all of the aforementioned models, the required capacity on an edge is the sum of the required capacities in both directions. Instead, Bienstock and Günlük [4], and Bienstock et al. [3] consider capacities which are bi-directional, i.e. the required capacity on an edge is the maximum of the required capacities in the two directions. All of the above models assume that the topology in the Backbone is already given and they do not treat the location problem.

Other papers focus on Local Access Networks (LANs) (see for instance Gavish [9], Balakrishnan et al. [1]). LANs are of particular interest since they account for a large portion of the total investments in telecommunication networks. Hence, even marginal improvements in the LAN design can lead to significant overall savings. Balakrishnan, Magnanti and Wong [2] use decomposition and polyhedral techniques to tackle the Local Access Network Expansion Problem, while Cho and Shaw [6] employ a column generation approach. Flippo et al. [7] show that the problem can be solved to optimality in pseudopolynomial time, by using a dynamic programming algorithm.

In this paper we discuss the ATM Network Installation Problem (denoted ANIP in the sequel) on a given tree, which consists of all the nodes in the area being served by the same Backbone node. The root of the tree is thus a node in the Backbone, in which an ATM hardware device is installed by default. Traffic demand between pairs of nodes in the tree are also given (these are

called commodities). To enable communication between the endpoints of a commodity, capacity has to be installed on the edges of a *walk* in the tree between the nodes involved. Note that a walk is defined here as a path in which edge repetition is allowed. The reason for this edge repetition will become clear in Section 2. By installing a hardware device (called an ATM cross connect) in a node, ATM signals can be combined such that the required capacity on the edges can be reduced. The key issue in ANIP is thus to find a trade-off between costs of capacity installation on edges on the one hand, and costs of installing ATM cross connects in nodes on the other hand, such that all demand is met and total costs are minimized.

In the sequel, ANIP is shown to be NP-hard, as it contains SUBSET SUM as a special case. We state a pseudo-polynomial time dynamic programming algorithm which runs in $\mathcal{O}(nB^4)$ and requires $\mathcal{O}(nB^2)$ storage space. Here, n refers to the number of nodes in the tree, and B to an upperbound on the capacity of an ATM cross connect. Although the algorithm can easily be applied when demand is non-symmetric, for ease of exposition we will assume throughout this paper that demand is indeed symmetric. Computational experiments indicate that our algorithm runs efficiently on real-life problem instances, made available to us by KPN Research, Leidschendam, The Netherlands.

The remainder of this paper is organized as follows. In Section 2 we give a detailed description of different aspects of the functionality of ATM networks and its components, for as far as they are relevant for the problem at hand. The notation used throughout this paper and a mathematical formulation of the problem are stated in Section 3. In Section 4 two families of subproblems are defined, and the relations between these subproblems are discussed. The dynamic programming algorithm and its use as an ATM network planning tool are the subject of Section 5. Computational results conclude the paper in Section 6.

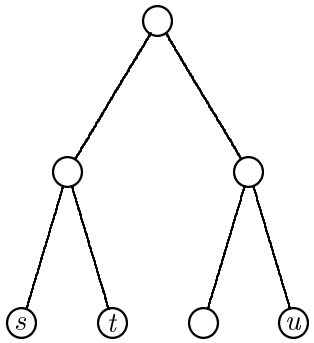
2 ATM functionality

In this section we give a detailed description of an ATM tree network and the hardware devices it may contain. We confine ourselves to those functionalities which are important for the problem at hand. An ATM tree network consists of ATM cross connects (to be referred to as concentrators henceforth) that can be installed in the nodes of the tree, and cables between these concentrators (usually referred to as connections). If a connection is installed between two concentrators, it actually requires capacity installation on each edge of the (unique) path between the end points of the connection. We consider two types of connections, viz. 34Mb/s and 155Mb/s connections (of course the model can be extended to incorporate other types of connections as well). The capacity of a connection is bi-directional, i.e. a 34Mb/s connection can accomodate 34Mb/s in both directions simultaneously. Connections can be used to transport information between demand nodes (nodes in the underlying tree structure) and concentrators on the one hand, and between concentrators themselves on the other. We assume that all nodes in the tree are demand nodes, that is each node in the tree is the source (and by symmetry of demand also the destination) of a commodity. If necessary, the demand of certain commodities can be set equal to zero.

A concentrator contains a given number of slots (cf. Figure 1(c)). In each of these slots an interface card of a certain type can be installed. As for connections, we consider two types of interface cards, viz. 34Mb/s and 155Mb/s interface cards. Each type of card has a number of input/output gates which equals the number of connections that can be connected to the interface card. Obviously, 34Mb/s (155Mb/s) connections can only be connected to 34Mb/s (155Mb/s) cards. Different types of concentrators are available, which for our purpose only differ in the number of slots they provide. This number of slots determines a concentrator's capacity. Apart from slots a concentrator also contains a routing table. An ATM cell that enters a concentrator via an input/output gate of one of the interface cards installed, is passed through the routing table, which evaluates the routing information in the cell to determine on which connection the cell should leave the concentrator. This gives the concentrator the capability to separate two (or more) commodities which enter the concentrator via the same connection by sending them out on different connections. Of course the reverse, i.e. bundling of commodities, is also possible.

The overall ATM tree network should now be constructed in such a way that each commodity is routed from its source node to its destination node. This route may pass several ATM cross connects on its way. A connection must be installed on this route in any of three cases : on the unique path between each pair of consecutive ATM cross connects on this route, between the source of a commodity and its nearest ATM cross connect on the route, and between the destination of the commodity and its nearest ATM cross connect. Furthermore, a sufficient number of interface cards should be installed in the ATM cross connects in order to connect the connections to the ATM cross connects. Since the complete route from source node to destination node for a commodity is thus a concatenation of connections, the route is often called a virtual path in telecommunication literature. However, since the concatenation of paths (which itself do not allow for edge repetition) implies that on the complete route itself edge repetition might occur when a specific edge occurs in more than one of these paths, from a graph theoretical point of view it should really be referred to as a walk (which by definition allows for edge repetition). To explain the above in more detail, consider the illustration in Figure 1.

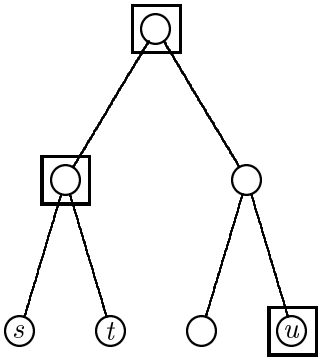
In order to enable communication between demand nodes s, t and u, all of these nodes must be connected to a concentrator. The concentrator to which a demand node is connected is called its "homing node" (or "homing concentrator"). Suppose that a concentrator is installed in the parent node of node s and t, serving as a homing node for nodes s and t, that a concentrator is installed in u which is the homing node for node u, and that a final concentrator is installed in the root. This configuration is depicted in Figure 1(b), where squared boxes indicate the installation of a concentrator. If the origin and destination of a commodity have a different homing node, a virtual walk (as explained in the above) between the two homing nodes must be installed. In general, this walk may pass other concentrators. Figure 1(c) shows what type of connections may be added to enable the desired communication between nodes s, t and u (the proposed configuration is not claimed to be optimal). Commodity (s, u) enters the homing concentrator of s on a 34Mb/s connection, and the same holds for commodity (t, u). Next, the two commodities are bundled and transferred to the concentrator in the root on a single 155Mb/s connection. There the commodities are simply passed on, again via a 155Mb/s connection, towards the concentrator in node u. This connection passes two edges in the underlying graph,



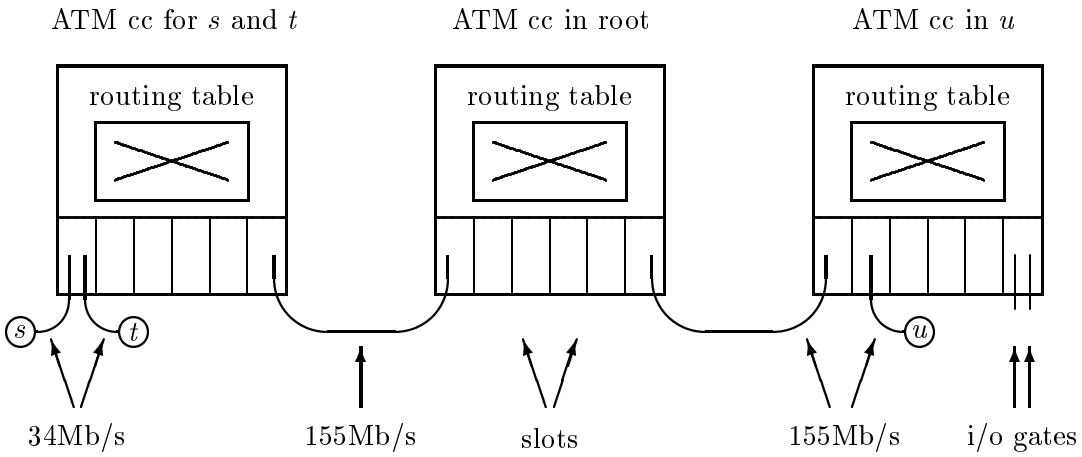
Demands in Mb/s :

	s	t	u
s	–	0	30
t	0	–	30
u	30	30	–

(a) problem instance



(b) concentrator installation



(c) detailed description

Figure 1: example instance

and therefore requires capacity installation on both edges. Finally the commodities are delivered at node u via a 155Mb/s connection. Note that to route the commodities from a demand node to its homing concentrator, and to separate/combine commodities at a demand node, some hardware device other than a concentrator must be installed at the demand node as well. In general, such hardware devices must be installed at every demand node. Therefore, these devices do not influence the design problem, and are left out of consideration. The reverse walk can be used for commodity (u, s) and (u, t). The reader may note that 34Mb/s (155Mb/s) interface cards have two (one) input/output gates. We will say more about the influence of the number of input/ouput gates in Section 3.

In the solution described in the above and depicted in Figure 1, no edge repetition occurs. Suppose however, that no concentrator was installed in the parent node of demand nodes s and t, and that the concentrator in the root node would serve as their homing node. Furthermore, assume that there would be a positive demand commodity to be routed from origin node s to destination node t. From the origin node s, the commodity would be routed to its homing node in the root of the tree (using the unique path between node s and the root node). Next, the commodity would be routed via the unique path between the root node and destination node t. The reader may observe that in this case the concatenation of these two paths does imply edge repetition on the complete route from origin to destination for the commodity, since the edge between the root node and the parent node of s and t is passed two times.

Network planners often impose certain restrictions on the layout of telecommunication networks. Some of these restrictions may be due to technical requirements, whereas others are considered to be economically sensible, or practicle for operational convenience regarding maintenance and repair. For ANIP, the following restrictions apply.

1. **Upward homing of demand nodes:** a demand node always homes on the first concentrator located on the unique path from the demand node to the root of the tree (both endpoints including);
2. **Upward homing of concentrators:** connections between two concentrators are only possible if one concentrator "homes" on the other, where a concentrator in node v (other than the root node) always homes on the first concentrator located on the path from (but excluding) v to the root. For example in Figure 1, the concentrator in node u is said to be homing on the concentrator in the root node, and the same holds for the concentrator located in the parent node of node s and t.

The reader may observe that the ATM network proposed in the example instance satisfies both routing restrictions.

Next, it needs to be stated that a commodity may not be split over different connections. This implies that 3 commodities, each with a demand of 20 Mb/s, cannot be routed on two 34Mb/s connections (although the total capacity seems to suffice), since it would require one of the commodities to be split over two connections. Define J_u as the set of possible connections between a demand node u and its homing concentrator. For example, if the largest capacity

concentrator contains 10 slots, and the number of input/ouput gates on a 34Mb/s and 155Mb/s interface card equals two and one, respectively, then J_u consists of twenty 34Mb/s connections and ten 155Mb/s connections. Furthermore, let D_u be the set of commodities which are to be sent from a demand node u to its homing concentrator (all in the same direction). Finally, define the following decision variables and parameters.

$$
\begin{array}{lcll}
m_u^j & = & \left\{ \begin{array}{ll} 1 & \text{if connection } j \text{ is used} \\ 0 & \text{otherwise} \end{array} \right. & (j \in J_u) \\
t_u^{ij} & = & \left\{ \begin{array}{ll} 1 & \text{if commodity } i \text{ is assigned to connection } j \\ 0 & \text{otherwise} \end{array} \right. & (i \in D_u,\ j \in J_u) \\
\delta_u^i & = & \text{demand of commodity } i & (i \in D_u) \\
cap_u^j & = & \text{capacity of connection } j & (j \in J_u)
\end{array}
$$

Then the collection of connections required to route the commodities of demand node u to its homing concentrator should satisfy the following restrictions:

$$
\begin{array}{lll}
\sum_{j \in J_u} t_u^{ij} = 1 & \forall i \in D_u & (1) \\
\sum_{i \in D_u} \delta_u^i \cdot t_u^{ij} \leq cap_u^j \cdot m_u^j & \forall j \in J_u & (2) \\
m_u^j \in \{0,1\},\ t_u^{ij} \in \{0,1\} & \forall i \in D_u,\ \forall j \in J_u & (3)
\end{array}
$$

Likewise, for all commodities which must be routed from a concentrator to its homing concentrator, the required collection of connections involved should satisfy a similar set of constraints. This problem is referred to as the bin-packing problem of commodities on connections. We will return to this issue in Section 3, after we have given a mathematical formulation of the problem.

The costs of an ATM network structure stem from concentrators and connections. The following assumptions are made regarding to the costs:

- the costs of a concentrator are fully determined by the number of connections of each type connected to it, and by the node in which it is installed;
- the costs of a connection equal the sum of the costs on the edges on the connection (path);
- the costs of the concentrators are independent of the costs of the connections.

Note that these assumptions allow for very general costs structures, and encompass the typical fixed-charge cost functions observed in practice. Now all ingredients for the ANIP are discussed in words, a formal description can be given.

3 Notation and Mathematical Formulation

We introduce the following problem parameters:

$G(\mathcal{V},\mathcal{E})$	=	tree G, where $\mathcal{V}$ represents the node set and $\mathcal{E}$ the undirected edge set; the nodes are numbered in a depth first order with the root numbered 0; the edges are also numbered in a depth first order such that edge v is the edge between node v and its unique parent
d_v	=	the number of children of node v; the children are denoted by $s_v^1, \ldots, s_v^{d_v}$
$T[v,i]$	=	subtree induced by v, its first i children and all successors of these children
$V[v,i]$	=	set of nodes in subtree $T[v,i]$
$E[v,i]$	=	set of edges in subtree $T[v,i]$
$V(u,w)$	=	set of all nodes on the path from u to w, including both endpoints
$E(u,w)$	=	set of all edges on the path from u to w
X_u	=	set of possible homing nodes for a demand node u, i.e. $X_u = V(u,0)$
Y_u	=	set of possible homing nodes for a concentrator located in u, i.e. $Y_u = V(u,0)\backslash\{u\}$
T	=	set of available capacities (interface cards), e.g. $T = \{34, 155\}$ let $t = \|T\|$ denote the cardinality of T
$K_w(k_w)$	=	costs of a concentrator located in w if k_w represents the number of connections that are connected to this concentrator ($k_w \in \mathbb{N}^t$); $K_w(0)$ corresponds to the situation where no concentrator is installed in node w
$L_e(\ell_e)$	=	costs on an edge $e \in \mathcal{E}$ if ℓ_e represents the number of connections over edge e ($\ell_e \in \mathbb{N}^t$)
γ	=	number of input/output gates on interface cards ($\gamma \in \mathbb{N}^t$)
B	=	number of slots available in the largest capacity concentrator
D_u	=	set of commodities with source node u
P_u	=	set of commodities with its source node located in $T[u,d_u]$ and with destination node outside $T[u,d_u]$
J_u	=	set of available connections between demand node u and its homing concentrator; let $J_u = \cup_{\tau \in T} J_u^\tau$
H_u	=	set of available connections between a concentrator in u (given that a concentrator is installed there) and its homing concentrator; let $H_u = \cup_{\tau \in T} H_u^\tau$
δ_u^i	=	demand of commodity $i \in D_u$
cap_u^j	=	capacity of connection $j \in J_u$ or H_u

Note that $T[v,i] = T[v,i-1] \cup T[s_v^i, d_{s_v^i}] \cup \{s_v^i, v\}$, which will be exploited throughout the rest of this paper frequently. Moreover, note that $P_u = \emptyset$ for $u = 0$, and define $H_u = \emptyset$, for $u = 0$. Next we introduce the following decision variables :

$$x_{uw} = \begin{cases} 1 & \text{if demand node } u \text{ is homing on a concentrator in node } w \\ 0 & \text{otherwise} \end{cases} \qquad (u \in \mathcal{V}, w \in X_u)$$

$$y_{uw} = \begin{cases} 1 & \text{if a concentrator in } u \text{ is homing on a concentrator in } w \\ 0 & \text{otherwise} \end{cases} \qquad (u \in \mathcal{V} \setminus \{0\}, w \in Y_u)$$

$$
\begin{array}{lcll}
m_u^j & = & \left\{ \begin{array}{ll} 1 & \text{if connection } j \text{ is used} \\ 0 & \text{otherwise} \end{array} \right. & (u \in \mathcal{V}, j \in J_u) \\
t_u^{ij} & = & \left\{ \begin{array}{ll} 1 & \text{if commodity } i \text{ is assigned to connection } j \\ 0 & \text{otherwise} \end{array} \right. & (u \in \mathcal{V}, i \in D_u, j \in J_u) \\
n_u^j & = & \left\{ \begin{array}{ll} 1 & \text{if connection } j \text{ is used} \\ 0 & \text{otherwise} \end{array} \right. & (u \in \mathcal{V}, j \in H_u) \\
z_u^{ij} & = & \left\{ \begin{array}{ll} 1 & \text{if commodity } i \text{ is assigned to connection } j \\ 0 & \text{otherwise} \end{array} \right. & (u \in \mathcal{V}, i \in P_u, j \in H_u) \\
k_w^\tau & = & \text{number of connections of type } \tau \text{ incident to node } w & (w \in \mathcal{V}, \tau = 1, \ldots, t) \\
\ell_e^\tau & = & \text{number of connections of type } \tau \text{ on edge } e & (e \in \mathcal{E}, \tau = 1, \ldots, t)
\end{array}
$$

The ANIP is now given by the following mathematical formulation :

$$
\begin{array}{lll}
\min & \sum_{w \in \mathcal{V}} K_w(k_w) + \sum_{e \in \mathcal{E}} L_e(\ell_e) & (4) \\
\text{s.t.} & \sum_{w \in X_u} x_{uw} = 1 & \forall u \in \mathcal{V} \quad (5) \\
& x_{u'w} \geq x_{uw} & \forall u, u', w \in \mathcal{V} : w \in Y_u, u' \in V(u,w) \backslash \{u\} \quad (6) \\
& \sum_{w \in Y_u} y_{uw} = x_{uu} & \forall u \in \mathcal{V} \setminus \{0\} \quad (7) \\
& y_{uw} \leq x_{u'w} & \forall u, u', w \in \mathcal{V} : w \in Y_u, u' \in V(u,w) \backslash \{u\} \quad (8) \\
& \sum_{j \in J_u} t_u^{ij} = 1 & \forall u \in \mathcal{V}, \forall i \in D_u \quad (9) \\
& \sum_{i \in D_u} \delta_u^i \cdot t_u^{ij} \leq cap_u^j \cdot m_u^j & \forall u \in \mathcal{V}, \forall j \in J_u \quad (10) \\
& \sum_{j \in H_u} z_u^{ij} = x_{uu} & \forall u \in \mathcal{V}, \forall i \in P_u \quad (11) \\
& \sum_{i \in D_u} \delta_u^i \cdot z_u^{ij} \leq cap_u^j \cdot n_u^j & \forall u \in \mathcal{V}, \forall j \in H_u \quad (12) \\
& k_w^\tau = \sum_{u \in \mathcal{V}} (\sum_{j \in J_u^\tau} m_u^j x_{uw} + \sum_{j \in H_u^\tau} n_u^j y_{uw}) & \\
& \qquad + \sum_{j \in H_w^\tau} n_w^j x_{ww} & \forall w \in \mathcal{V}, \tau = 1, \ldots, t \quad (13) \\
& \ell_e^\tau = \sum_{u,w \in \mathcal{V} : e \in E(u,w)} (\sum_{j \in J_u^\tau} m_u^j x_{uw} + \sum_{j \in H_u^\tau} n_u^j y_{uw}) & \\
& & \forall e \in \mathcal{E}, \tau = 1, \ldots, t \quad (14) \\
& x_{uw} \in \{0,1\} & \forall u \in \mathcal{V}, \forall w \in X_u \quad (15) \\
& y_{uw} \in \{0,1\} & \forall u \in \mathcal{V} \setminus \{0\}, \forall w \in Y_u \quad (16) \\
& k_w,\ \ell_e \geq 0 & \forall w \in \mathcal{V}, \forall e \in \mathcal{E} \quad (17) \\
& m_u^j,\ n_u^j,\ t_u^{ij},\ z_u^{ij} \in \{0,1\} & \forall u \in \mathcal{V}, \forall i \in D_u, P_u,\ \forall j \in J_u, H_u \quad (18)
\end{array}
$$

The objective function in (4) defines the total costs for a solution (x, y, m, t, n, z, k, l) and reflects the decomposability of the costs on concentrators installed in nodes and connections passing over

edges. Constraint (5) implies that every demand node has exactly one homing node, and by the definition of X_u, it follows that a concentrator is installed in the root node. Constraint (6) enforces the upward homing condition of demand nodes, whereas constraint (7) and (8) guarantee the upward homing of concentrators. The bin-packing problem of commodities on connections is modelled by constraints (9) and (10) for a demand node and its homing concentrator, and by constraints (11) and (12) for commodities between a concentrator and its homing concentrator, respectively. Constraints (13) and (14) simply define the resulting number of connections connected to a concentrator in a node, and the number of connections which pass over an edge, respectively. The integrality and non-negativity constraints in (15)–(18) complete the formulation. Note that we have not listed a constraint which restricts the number of connections incident to a concentrator, which could have been formulated as

$$\sum_{\tau=1}^{t} \lceil k_w^\tau / \gamma^\tau \rceil \leq B \qquad \forall w \in \mathcal{V} \tag{19}$$

For ease of exposition, (19) will be stated as $k_w \ll B$ in the sequel, and should be interpreted as "the number of connections k_w fits on the largest capacity concentrator". Note that if a given number of connections k_w would require more slots than there are available in the largest capacity concentrator, this infeasibility can easily be incorporated in the objective function by setting the corresponding $K_w(k_w)$ equal to infinity. Hence, constraint (19) is not part of our model. In Section 5 however, the number B will be shown to be an important determinant for the running time of our algorithm.

The objective function (4) may encompass very general costs structures. For the typical fixed cost structure arising in practice, the objective function can be specified as follows. Let θ refer to the different types of concentrators available, B_θ denote its capacity, and $\hat{F}_w(\theta)$ represent the fixed costs of installing a concentrator of type θ in node w, and μ_w^τ be the costs of an interface card of type τ in node w. Then

$$K_w(k_w) \quad = \quad \begin{cases} \min_{\theta: k_w \ll B_\theta} \{\hat{F}_w(\theta) + \sum_{\tau=1}^{t} \mu_w^\tau \cdot \lceil k_w^\tau / \gamma^\tau \rceil\} & k_w \neq 0 \\ 0 & k_w = 0 \end{cases} \tag{20}$$

with the minimum over an empty set defined as infinity. Throughout this paper, we will assume that large capacity concentrators are available (since this is also the case in all real-life problem instances), which implies that a feasible solution always exists. Moreover, if λ_e^τ denotes the costs of installing a connection of type τ on an individual edge e, then

$$L_e(\ell_e) \quad = \quad \sum_{\tau=1}^{t} \lambda_e^\tau \cdot \ell_e^\tau \tag{21}$$

The reader may notice that for a given choice of the variables x_{uw}, y_{uw}, the remaining variables $t_u^{ij}, z_u^{ij}, m_u^j, n_u^j$ only interact with the previous mentioned variables via the objective function,

since constraints (13) and (14) merely define the variables k_w and ℓ_e. For general objective functions (as well as parameter values), this interaction may of course imply that the optimal values of the two classes of decision variables cannot be determined independently. Next, we derive conditions (which are fulfilled in the practical problem instances obtained from KPN Research, Leidschendam, The Netherlands), under which the number of connections used between a demand node and its homing concentrator (variables t_u^{ij}, m_u^j), as well as the number of connections used between a concentrator and its homing concentrator (variables z_u^{ij}, n_u^j), can be determined optimally beforehand, that is, independently of the choice of the remaining variables.

Lemma 3.1 *Consider ANIP with objective function as defined by (4), (20) and (21), and with $t = 2$, i.e. two types of connections (interface cards), viz. 34Mb/s and 155Mb/s. If*

- *the interface card costs are independent of its type and positive, i.e. $\mu_w^\tau = \mu_w > 0$ for all τ and for all $w \in \mathcal{V}$*
- *the costs of installing two 34Mb/s connections on an edge are at least the costs of installing one 155Mb/s connection on an edge, and positive, i.e. $2 \cdot \lambda_e^{34} \geq \lambda_e^{155} > 0$, for all $e \in \mathcal{E}$*
- *the number of input/output gates on a 34Mb/s and 155Mb/s interface card is two and one, respectively, i.e. $\gamma^{34} = 2, \gamma^{155} = 1$*

then there exists an optimal solution (x, y, m, t, n, z, k, l) for ANIP in which each demand node is connected to its homing concentrator using at most one 34Mb/s connection, i.e. $\sum_{j \in J_u^{34}} m_u^j \leq 1$, $\forall u \in \mathcal{V}$, and each concentrator is connected with its homing concentrator using at most one 34Mb/s connection, i.e. $\sum_{j \in H_u^{34}} n_u^j \leq 1$, $\forall u \in \mathcal{V} \setminus \{0\}$.

Proof. Suppose we are given a solution for ANIP in which a demand node u is connected to its homing concentrator using more than one 34Mb/s connection. Then replacing two of these 34Mb/s connections by one 155Mb/s connection, as well as replacing a 34Mb/s interface card by a 155Mb/s interface card in the concentrator, again yields a feasible solution, for which the costs have not increased. Repeating this argument yields the first part of the result. The second part of our claim can be verified similarly. ■

Lemma 3.2 *(i). Consider the bin-packing problem for a demand node u as defined by constraints (9) and (10), and with objective function*

$$min \quad \sum_{j \in J_u} \lambda_j \cdot m_u^j \tag{22}$$

with $2 \cdot \lambda^{34} \geq \lambda^{155} > 0$. Let $(\tilde{t}, \tilde{m})$ represent an optimal solution for this bin-packing problem. If the conditions of Lemma 3.1 are satisfied, then there exists an optimal solution $(\bar{x}, \bar{y}, \bar{k}, \bar{l}, \bar{t}, \bar{m}, \bar{z}, \bar{n})$ for ANIP with $\bar{t} = \tilde{t}, \bar{m} = \tilde{m}$.

(ii). Consider the bin-packing problem for a potential concentrator node $u \neq 0$ as defined by constraints (11) and (12), and with objective function

$$min \quad \sum_{j \in H_u} \lambda_j \cdot n_u^j \tag{23}$$

with $2 \cdot \lambda^{34} \geq \lambda^{155} > 0$. Let $(\tilde{z}, \tilde{n})$ represent an optimal solution for this bin-packing problem. If the conditions of Lemma 3.1 are satisfied, and there exists an optimal solution for ANIP with a concentrator installed in node u, then there exists an optimal solution $(\bar{x}, \bar{y}, \bar{k}, \bar{l}, \bar{t}, \bar{m}, \bar{z}, \bar{n})$ for ANIP with $\bar{z} = \tilde{z}, \bar{n} = \tilde{n}$.

Proof. Let $(\tilde{t}, \tilde{m})$ represent an optimal solution of the bin-packing problem and $(\bar{x}, \bar{y}, \bar{k}, \bar{l}, \bar{t}, \bar{m}, \bar{z}, \bar{n})$ be an optimal solution for ANIP. Now suppose that $\tilde{m} \neq \bar{m}$. Define

$$\tilde{b}_u^\tau = \sum_{j \in J_u^\tau} \tilde{m}_u^j$$

to be the number of connections of type τ used in the optimal bin-packing solution, and let $\bar{b}_u^\tau$ be defined similarly to be the number of connections of type τ used in the optimal ANIP solution. We consider 8 cases.

(i). $\tilde{b}_u^{34} = \bar{b}_u^{34}, \tilde{b}_u^{155} > \bar{b}_u^{155}$; then a simple exchange argument makes it is easy to verify that $\tilde{m}$ cannot be an optimal solution for the bin-packing problem.

(ii). $\tilde{b}_u^{34} = \bar{b}_u^{34}, \tilde{b}_u^{155} < \bar{b}_u^{155}$; then $\bar{m}$ is not an optimal solution for ANIP.

(iii). $\tilde{b}_u^{34} > \bar{b}_u^{34}, \tilde{b}_u^{155} = \bar{b}_u^{155}$; then $\tilde{m}$ is not an optimal solution for the binpacking problem.

(iv). $\tilde{b}_u^{34} < \bar{b}_u^{34}, \tilde{b}_u^{155} = \bar{b}_u^{155}$; then $\bar{m}$ is not an optimal solution for ANIP.

(v). $\tilde{b}_u^{34} > \bar{b}_u^{34}, \tilde{b}_u^{155} > \bar{b}_u^{155}$; then $\tilde{m}$ is not an optimal solution for the binpacking problem.

(vi). $\tilde{b}_u^{34} < \bar{b}_u^{34}, \tilde{b}_u^{155} < \bar{b}_u^{155}$; then $\bar{m}$ is not an optimal solution for ANIP.

(vii). $\tilde{b}_u^{34} > \bar{b}_u^{34}, \tilde{b}_u^{155} < \bar{b}_u^{155}$; from Lemma 3.1 it follows that we may assume w.l.o.g. that $\tilde{b}_u^{34} = 1, \bar{b}_u^{34} = 0$. Next, if $\bar{b}_u^{155} \geq \tilde{b}_u^{155} + 2$, then $\bar{b}_u$ cannot be optimal for ANIP, so assume that $\bar{b}_u^{155} = \tilde{b}_u^{155} + 1$. But then it holds that replacing the connections $\bar{m}_u$ by connections $\tilde{m}_u$ yields a feasible solution for ANIP with lower costs, hence $\bar{m}_u$ is not optimal for ANIP.

(viii). $\tilde{b}_u^{34} < \bar{b}_u^{34}, \tilde{b}_u^{155} > \bar{b}_u^{155}$; from Lemma 3.1 it follows that we may assume w.l.o.g. that $\tilde{b}_u^{34} = 0, \bar{b}_u^{34} = 1$. Next, if $\bar{b}_u^{155} \leq \tilde{b}_u^{155} - 2$, then $\tilde{b}_u$ cannot be optimal for the bin-packing problem, so assume that $\bar{b}_u^{155} = \tilde{b}_u^{155} - 1$. But then it holds that replacing the connections $\tilde{m}_u$ by connections $\bar{m}_u$ yields a feasible solution for the bin-packing problem with lower costs, hence $\tilde{m}_u$ is not optimal for the bin-packing problem.

This shows that indeed $\bar{m}_u = \tilde{m}_u$ must hold. Its also easy to see that $\bar{t} = \tilde{t}$, since the same assignment of the commodities can be used for both problems. Similarly, one can prove the second part of the Lemma. ■

Under the conditions of Lemma 3.1, the optimal values of the variables $t_u^{ij}, z_u^{ij}, m_u^j, n_u^j$ can thus be determined independently of the homing and routing decisions, by using the bin-packing problems as defined in Lemma 3.2. Hence, the optimal number of connections between a demand node u and its homing concentrator (denoted b_u) can be obtained beforehand (although it requires solving an NP-hard problem):

$$b_u^\tau = \sum_{j \in J_u^\tau} m_u^j$$

with m_u^j the optimal value of the corresponding bin-packing problem. Similarly, the optimal number of connections between a concentrator and its homing concentrator (denoted p_u) can be determined beforehand :

$$p_u^\tau = \sum_{j \in H_u^\tau} n_u^j$$

with n_u^j the optimal value of the corresponding bin-packing problem, and the sum over an empty set defined equal to zero. Thus, under the conditions of Lemma 3.2 ANIP can be formulated as follows (denoted ANIP* in the sequel):

$$\begin{array}{lll}
\min & \sum_{w \in \mathcal{V}} K_w(k_w) + \sum_{e \in \mathcal{E}} L_e(\ell_e) & (24) \\
& \sum_{w \in X_u} x_{uw} = 1 & \forall u \in \mathcal{V} \quad (25) \\
& x_{u'w} \geq x_{uw} & \forall u, u', w \in \mathcal{V} : w \in X_u, u' \in V(u,w) \backslash \{u\} \quad (26) \\
& \sum_{w \in Y_u} y_{uw} = x_{uu} & \forall u \in \mathcal{V} \setminus \{0\} \quad (27) \\
& y_{uw} \leq x_{u'w} & \forall u, u', w \in \mathcal{V} : w \in Y_u, u' \in V(u,w) \backslash \{u\} \quad (28) \\
& k_w^\tau = \sum_{u \in \mathcal{V}} (b_u^\tau x_{uw} + p_u^\tau y_{uw}) + p_w^\tau x_{ww} & \forall w \in \mathcal{V}, \tau = 1, \ldots, t \quad (29) \\
& \ell_e^\tau = \sum_{u,w \in \mathcal{V} : e \in E(u,w)} (b_u^\tau x_{uw} + p_u^\tau y_{uw}) & \forall e \in \mathcal{E}, \tau = 1, \ldots, t \quad (30) \\
& x_{uw} \in \{0,1\} & \forall u \in \mathcal{V}, \forall w \in X_u \quad (31) \\
& y_{uw} \in \{0,1\} & \forall u \in \mathcal{V} \setminus \{0\}, \forall w \in Y_u \quad (32) \\
& k_w^\tau,\ \ell_e^\tau \geq 0 & \forall w \in \mathcal{V},\ \forall e \in \mathcal{E},\ \tau = 1, \ldots, t \quad (33)
\end{array}$$

Unless stated otherwise, in the sequel we will assume that the conditions of Lemma 3.2 are satisfied, hence the latter formulation of ANIP* will be used.

Now we show that the decision version of ANIP* is NP-complete, which implies that an exact algorithm running in pseudo-polynomial time is the best one may expect, unless P=NP. To do so, we state the following problem definitions:

SUBSET SUM (see Garey and Johnson [8])
INSTANCE: Finite set A, size $s(a) \in \mathbb{N}$ for every $a \in A$, and a positive integer D.
QUESTION: Is there a subset $A' \subseteq A$ such that the sum of the sizes of the elements in A' is equal to D?

ANIP*
INSTANCE: Tree $G = (\mathcal{V}, \mathcal{E})$, an integer B, a number $t \in \mathbb{N}$, a demand $b_u^\tau \in \mathbb{N}$ for every $u \in \mathcal{V}, \tau = 1, \ldots, t$, a number $p_u \in \mathbb{N}$ for every $u \in \mathcal{V}, \tau = 1, \ldots, t$, and cost functions $K_w(k_w) : \Pi_\tau\{0, \ldots, \gamma^\tau \cdot B\} \to \mathbb{N}$, for all $w \in \mathcal{V}$ and $L_e(\ell_e) : \Pi_\tau\{0, \ldots, \gamma^\tau \cdot B\} \to \mathbb{N}$, for all $e \in \mathcal{E}$, and a positive integer F.
QUESTION: Does there exist a solution (x, y, k, ℓ) for ANIP as defined by (24)–(33) with objective value at most F?

Theorem 3.1 *ANIP* is NP-complete (assuming that for each k_w, ℓ_e and F, $\sum_w K_w(k_w) + \sum_e L_e(\ell_e) \leq F$ can be verified in polynomial time).*

Proof. It can easily be checked that ANIP* is in NP (given the mild assumption that the objective value can be verified in polynomial time). Hence it suffices to show that SUBSET SUM reduces to ANIP*. Given an instance for SUBSET SUM, we define an instance for ANIP* as follows. Let $\mathcal{V} = A \cup \{0\}$ contain a node for each item in A and a node which will be the root of the tree. Let $\mathcal{E} = \{\{0, a\} | a \in A\}$ be the set of edges where each node in A is connected to the root. Next, we let $t = 1$ and $\gamma^t = 1$ (only one type of connection and the number of slots equals the number of connections which can be connected to a concentrator). Define $b_a = s(a)$ for all $a \in A$ and $b_0 = 0$, $p_a = 0$ for all $a \in A \cup \{0\}$, $B = \sum_{a \in A} s(a)$, and $F = 1$. Finally, edge costs $L_e(\ell_e) = 0$ for all $e \in \mathcal{E}$, and $K_w(k_w) = 0$ for $w \neq 0$, $K_w(k_w) = 1$ for $w = 0, k_w = D$, and $K_w(k_w) = 2$ for $w = 0, k_w \neq D$. Note that this transformation can be done in polynomial time. It remains to show that the two problem instances are equivalent.

First, assume that there exists a subset $A' \subseteq A$ with an aggregated size equal to D. By installing a large concentrator in the nodes corresponding to items in $A \setminus A'$, as well as in the root, we obtain a feasible solution for ANIP* with objective value equal to one. Conversely, if there exists a solution for ANIP* with solution value less than or equal to one, it must be equal to one, since the concentrator costs in the root are at least one. But then the load on the concentrator in the root must equal D, which in turn implies that the set of nodes in which no concentrator is installed has a cumulative demand equal to D. ■

4 Defining Subproblems for ANIP

In this section we introduce two families of subproblems which are defined on subtrees $T[v,i]$. These subtrees were introduced by Johnson and Niemi [12] to improve the running time of a dynamic programming algorithm for partially ordered knapsack problems. The main idea is that a subtree $T[v,i]$ can be decomposed into two smaller subtrees $T[v,i-1]$ and $T[s_v^i, d_{s_v^i}]$ (provided $i \geq 1$). By combining optimal solutions for problems defined on the latter subtrees, one may be able to obtain an optimal solution for the former. The main difficulty (as is the case for all dynamic programming algorithms) is to find an appropriate parametrization of the problem at hand.

The first family of subproblems we define are denoted by $g(v,i,s)$. They represent the minimal costs restricted to the subtree $T[v,i]$, among all solutions in which a concentrator is installed in node v. Furthermore, the vector $s \in \mathbb{N}^t$ denotes the connections between the concentrator in node v and nodes in $V[v,d_v] \setminus V[v,i]$ (see Figure 2(a)). More formally, for (v,i) with $v \in \mathcal{V}$, $0 \leq i \leq d_v$, $s \geq 0$ and $s + b_v + p_v \ll B$ we define $g(v,i,s)$ to be

$$\min \quad \sum_{w \in V[v,i] \setminus \{v\}} K_w(k_w) + K_v(k_v + s) + \sum_{e \in E[v,i]} L_e(\ell_e) \tag{34}$$

$$\text{s.t.} \quad \sum_{w \in X_u \cap V[v,i]} x_{uw} = 1 \qquad \forall u \in V[v,i] \tag{35}$$

$$x_{u'w} \geq x_{uw} \qquad \forall u \in V[v,i], \forall w \in X_u \cap V[v,i], \forall u' \in V(u,w) \setminus \{u\} \tag{36}$$

$$\sum_{w \in Y_u \cap V[v,i]} y_{uw} = x_{uu} \qquad \forall u \in V[v,i] \setminus \{v\} \tag{37}$$

$$y_{uw} \leq x_{u'w} \qquad \forall u \in V[v,i], \forall w \in Y_u \cap V[v,i], \forall u' \in V(u,w) \setminus \{u\} \tag{38}$$

$$k_w^\tau = \sum_{u \in V[v,i]} (b_u^\tau x_{uw} + p_u^\tau y_{uw}) + p_w^\tau x_{ww} \qquad \forall w \in V[v,i], \tau = 1, \ldots, t \tag{39}$$

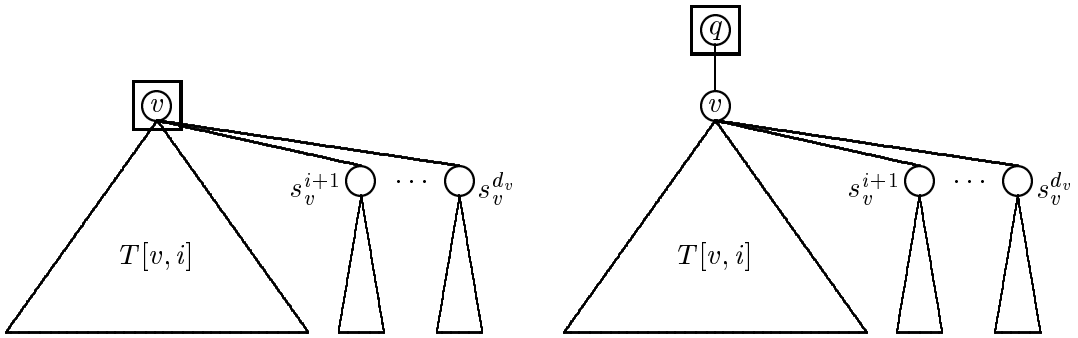
$$\ell_e^\tau = \sum_{u,w \in V[v,i]: e \in E(u,w)} (b_u^\tau x_{uw} + p_u^\tau y_{uw}) \qquad \forall e \in E[v,i], \tau = 1, \ldots, t \tag{40}$$

$$x_{uw} \in \{0,1\} \qquad \forall u \in V[v,i], \forall w \in V[v,i] \cap X_u \tag{41}$$

$$y_{uw} \in \{0,1\} \qquad \forall u \in V[v,i] \setminus \{v\}, \forall w \in V[v,i] \cap Y_u \tag{42}$$

$$k_w^\tau,\ \ell_e^\tau \geq 0 \qquad \forall w \in V[v,i], \forall e \in E[v,i], \tau = 1, \ldots, t \tag{43}$$

The second family of subproblems $h(v,i,r)$ represents the minimal costs restricted to the subtree $T[v,i]$, among all solutions for which no concentrator is installed in node v. The vector r represents the connections between nodes in $T[v,i]$ and a concentrator located outside $T[v,i]$. In order to have such a homing node for node v (as well as the endpoint outside $T[v,i]$ for the connections of the vector r), we introduce an artificial node q to be a predecessor of node v, in which a concentrator is installed (see Figure 2(b)). More formally, for (v,i) with $v \in \mathcal{V}$, $0 \leq i \leq d_v$ and $b_v \leq r \ll B$ we define $h(v,i,r)$ to be



(a) g(v,i,s) (b) h(v,i,r)

Figure 2: subproblems

$$\min \quad \sum_{w\in V[v,i]} K_w(k_w) + \sum_{e\in E[v,i]} L_e(\ell_e) \tag{44}$$

$$\text{s.t.} \quad \sum_{w\in X_u\cap V[v,i]} x_{uw} + x_{uq} = 1 \qquad \forall u \in V[v,i]\cup\{q\} \tag{45}$$

$$x_{vv} = 0 \tag{46}$$

$$x_{u'w} \geq x_{uw} \qquad \forall u \in V[v,i], \forall w \in (X_u \cap V[v,i])\cup\{q\}, \; \forall u' \in V(u,w)\setminus\{u\} \tag{47}$$

$$\sum_{w\in Y_u\cap V[v,i]} y_{uw} + y_{uq} = x_{uu} \qquad \forall u \in V[v,i] \tag{48}$$

$$y_{u'w} \leq x_{uw} \qquad \forall u \in V[v,i], \forall w \in (Y_u \cap V[v,i])\cup\{q\}, \; \forall u' \in V(u,w)\setminus\{u\} \tag{49}$$

$$k_w^\tau = \sum_{u\in V[v,i]}(b_u^\tau x_{uw} + p_u^\tau y_{uw}) + p_w^\tau x_{ww} \qquad \forall w \in V[v,i]\cup\{q\}, \tau = 1,\ldots,t \tag{50}$$

$$\ell_e^\tau = \sum_{u,w\in V[v,i]\cup\{q\}:e\in E(u,w)}(b_u^\tau x_{uw} + p_u^\tau y_{uw}) \qquad \forall e \in E[v,i]\cup\{v\}, \tau = 1,\ldots,t \tag{51}$$

$$\ell_v^\tau = r^\tau \qquad \tau = 1,\ldots,t \tag{52}$$

$$x_{uw} \in \{0,1\} \qquad \forall u \in V[v,i], \forall w \in (V[v,i]\cap X_u)\cup\{q\} \tag{53}$$

$$y_{uw} \in \{0,1\} \qquad \forall u \in V[v,i], \forall w \in (V[v,i]\cap Y_u)\cup\{q\} \tag{54}$$

$$k_w^\tau,\ \ell_e^\tau \geq 0 \qquad \forall w \in V[v,i], \forall e \in E[v,i], \tau = 1,\ldots,t \tag{55}$$

During our dynamic programming algorithm, solutions for subtree $T[v,i-1]$ and $T[s_v^i, d_{s_v^i}]$ will be combined to obtain a solution for subtree $T[v,i]$. We will therefore focus on the relations

between the subproblems we have just defined. In Section 4.1 we show how the problem $g(v,i,s)$ for $i > 0$ can be decomposed to problems on subtrees $T[v,i-1]$ and $T[s_v^i, d_{s_v^i}]$, whereas Section 4.2 reports on similar relations for $h(v,i,r)$. In Section 4.3 it is shown how $g(v,i,s)$ and $h(v,i,r)$ should be determined in case $i = 0$, i.e. if the subtree consists of a single node. These relations form the starting point of the dynamic programming algorithm for ANIP*.

4.1 Recursive relations for $g(v,i,s)$.

The set of feasible solutions for $g(v,i,s)$ can be partitioned into a set of solutions for which $x_{s_v^i s_v^i} = 1$ (representing the situation in which a concentrator is installed in node s_v^i), and a set of solutions for which $x_{s_v^i s_v^i} = 0$ (no concentrator is installed in node s_v^i). Let $\alpha \in \mathbb{N}^t$ denote the number of connections between a concentrator in node v and nodes in $V[v,i] \setminus V[v,i-1]$ in a feasible solution for ANIP with $x_{vv} = 1$, hence for $g(v,i,s)$. For a feasible solution in the case that $x_{s_v^i s_v^i} = 1$, α must equal $p_{s_v^i}$. The set of feasible solutions in the case that $x_{s_v^i s_v^i} = 0$, can be partitioned into smaller sets by considering all possible values of α. Because of the partition, the optimal value for $g(v,i,s)$ can be determined as the minimum of the optimal values over all subsets of feasible solutions. Therefore, one can show that the optimal value for $g(v,i,s)$ can be obtained from the aforementioned subsets of feasible solutions.

Lemma 4.1 *Consider (v,i) with $v \in \mathcal{V}$ and $i > 0$. Let (x,y,k,ℓ) be an optimal solution for $g(v,i,s)$ with $x_{s_v^i s_v^i} = 1$. Then*

$$g(v,i,s) \quad = \quad g(s_v^i, d_{s_v^i}, 0) + g(v, i-1, s + p_{s_v^i}) + L_{s_v^i}(p_{s_v^i}) \tag{56}$$

Proof. From $x_{s_v^i s_v^i} = 1$ it follows that $x_{uv} = 0$ for all $u \in V[s_v^i, d_{s_v^i}]$, and $y_{uv} = 0$ for all $u \in V[s_v^i, d_{s_v^i}] \setminus \{s_v^i\}$. But then constraints (35)–(43) of $g(v,i,s)$ can be formulated as follows:

$$\begin{aligned}
&\textstyle\sum_{w \in X_u \cap V[v,i-1]} x_{uw} = 1 && \forall u \in V[v,i-1] \\
&\textstyle\sum_{w \in X_u \cap V[s_v^i, d_{s_v^i}]} x_{uw} = 1 && \forall u \in V[s_v^i, d_{s_v^i}] \\
&x_{u'w} \geq x_{uw} && \forall u \in V[v,i-1], \forall w \in X_u \cap V[v,i-1] \\
& && \forall u' \in V(u,w) \setminus \{u\} \\
&x_{u'w} \geq x_{uw} && \forall u \in V[s_v^i, d_{s_v^i}], \forall w \in X_u \cap V[s_v^i, d_{s_v^i}] \\
& && \forall u' \in V(u,w) \setminus \{u\} \\
&\textstyle\sum_{w \in Y_u \cap V[v,i-1]} y_{uw} = x_{uu} && \forall u \in V[v,i-1] \setminus \{v\} \\
&\textstyle\sum_{w \in Y_u \cap V[s_v^i, d_{s_v^i}]} y_{uw} = x_{uu} && \forall u \in V[s_v^i, d_{s_v^i}] \setminus \{s_v^i\} \\
&y_{s_v^i v} = x_{s_v^i s_v^i} = 1 && \\
&y_{uw} \leq x_{u'w} && \forall u \in V[v,i-1], \forall w \in Y_u \cap V[v,i-1]
\end{aligned}$$

	$\forall u' \in V(u,w) \setminus \{u\}$
$y_{uw} \leq x_{u'w}$	$\forall u \in V[s_v^i, d_{s_v^i}], \forall w \in Y_u \cap V[s_v^i, d_{s_v^i}]$
	$\forall u' \in V(u,w) \setminus \{u\}$
$k_w^\tau = \sum_{u \in V[v,i-1]} (b_u^\tau x_{uw} + p_u^\tau y_{uw}) + p_w^\tau x_{ww}$	$\forall w \in V[v,i-1] \setminus \{v\}, \tau = 1, \ldots, t$
$k_w^\tau = \sum_{u \in V[s_v^i, d_{s_v^i}]} (b_u^\tau x_{uw} + p_u^\tau y_{uw}) + p_w^\tau x_{ww}$	$\forall w \in V[s_v^i, d_{s_v^i}], \tau = 1, \ldots, t$
$k_v^\tau = \sum_{u \in V[v,i-1]} (b_u^\tau x_{uw} + p_u^\tau y_{uw}) + p_v^\tau x_{vv} + p_{s_v^i}^\tau$	$\tau = 1, \ldots, t$
$\ell_e^\tau = \sum_{u,w \in V[v,i-1]: e \in E(u,w)} (b_u^\tau x_{uw} + p_u^\tau y_{uw})$	$\forall e \in E[v,i-1], \tau = 1, \ldots, t$
$\ell_e^\tau = \sum_{u,w \in V[s_v^i, d_{s_v^i}]: e \in E(u,w)} (b_u^\tau x_{uw} + p_u^\tau y_{uw})$	$\forall e \in E[s_v^i, d_{s_v^i}], \tau = 1, \ldots, t$
$\ell_{s_v^i}^\tau = p_{s_v^i}^\tau$	$\tau = 1, \ldots, t$
$x_{uw} \in \{0,1\}$	$\forall u \in V[v,i-1],$
	$\forall w \in V[v,i-1] \cap X_u$
$x_{uw} \in \{0,1\}$	$\forall u \in V[s_v^i, d_{s_v^i}],$
	$\forall w \in V[s_v^i, d_{s_v^i}] \cap X_u$
$y_{uw} \in \{0,1\}$	$\forall u \in V[v,i-1] \setminus \{v\},$
	$\forall w \in V[v,i-1] \cap Y_u$
$y_{uw} \in \{0,1\}$	$\forall u \in V[s_v^i, d_{s_v^i}] \setminus \{s_v^i\},$
	$\forall w \in V[s_v^i, d_{s_v^i}] \cap Y_u$
$k_w^\tau,\ \ell_e^\tau \geq 0$	$\forall w \in V[v,i-1],$
	$\forall e \in E[v,i-1] \tau = 1, \ldots, t$
$k_w^\tau,\ \ell_e^\tau \geq 0$	$\forall w \in V[s_v^i, d_{s_v^i}],$
	$\forall e \in E[s_v^i, d_{s_v^i}] \tau = 1, \ldots, t$

This shows that the constraints can be separated into two sets, one set containing the constraints on the subtree $T[v,i-1]$, and one set containing the constraints on the subtree $T[s_v^i, d_{s_v^i}]$. Since the objective function is also separable, it directly follows that the problem $g(v,i,s)$ decomposes into the problem $g(s_v^i, d_{s_v^i}, 0)$ on subtree $T[s_v^i, d_{s_v^i}]$, problem $g(v,i-1,s+p_{s_v^i})$ on subtree $T[v,i-1]$, and the individual edge s_v^i for which $\ell_{s_v^i} = p_{s_v^i}$. ■

Lemma 4.2 *Consider (v,i) with $v \in \mathcal{V}$ and $i > 0$. Let (x,y,k,ℓ) be an optimal solution for $g(v,i,s)$ with $x_{s_v^i s_v^i} = 0$ and $\ell_{s_v^i} = \alpha$. Then*

$$g(v,i,s) \quad = \quad h(s_v^i, d_{s_v^i}, \alpha) + g(v,i-1,s+\alpha) + L_{s_v^i}(\alpha) \tag{57}$$

Proof. Similar. ■

4.2 Recursive relations for $h(v,i,r)$.

In this section we use a similar partition of the set of feasible solutions for $h(v,i,r)$ to obtain its optimal value and solution. Since the proofs of the lemmas are similar to the one in the previous section, they are left to the reader.

Lemma 4.3 *Consider (v,i) with $v \in \mathcal{V}$ and $i > 0$. Let (x,y,k,ℓ) be an optimal solution for $h(v,i,r)$ with $x_{s_v^i s_v^i} = 1$. Then*

$$h(v,i,r) \;=\; g(s_v^i, d_{s_v^i}, 0) + h(v, i-1, r - p_{s_v^i}) + L_{s_v^i}(p_{s_v^i}) \tag{58}$$

Lemma 4.4 *Consider (v,i) with $v \in \mathcal{V}$ and $i > 0$. Let (x,y,k,ℓ) be an optimal solution for $h(v,i,r)$ with $x_{s_v^i s_v^i} = 0$ and $\ell_{s_v^i} = \alpha$. Then*

$$h(v,i,r) \;=\; h(s_v^i, d_{s_v^i}, \alpha) + h(v, i-1, r-\alpha) + L_{s_v^i}(\alpha) \tag{59}$$

4.3 Starting point of dynamic programming algorithm.

In this section we show how the coefficients $g(v,i,s)$ and $h(v,i,r)$ can be determined if $i = 0$, i.e. if the tree $T[v,i]$ consists of a single node.

Proposition 4.1 *Consider (v,i) with $v \in \mathcal{V}$ and $i = 0$. For $s \in \mathbb{N}^t$ such that $s + b_v + p_v \ll B$ it holds that*

$$g(v,i,s) = K_v(s + b_v + p_v) \tag{60}$$

Proof. It is easy to see that under the condition posed in the proposition there exists a solution for $g(v,0,s)$. Since $V[v,0] = \{v\}$, it follows that $x_{vv} = 1$. Furthermore,

$$\begin{aligned} k_v^\tau &= \textstyle\sum_{u \in V[v,0]} (b_u^\tau \cdot x_{uv} + p_u^\tau \cdot y_{uv}) + p_v^\tau \cdot x_{vv} \\ &= b_v^\tau \cdot x_{vv} + p_v^\tau \cdot x_{vv} \\ &= b_v^\tau + p_v^\tau \end{aligned}$$

The objective function in (34) thus amounts to $K_v(s + b_v + p_v)$. ■

Proposition 4.2 *Consider (v,i) with $v \in \mathcal{V}$ and $i = 0$. Then*

$$h(v,i,r) = \begin{cases} K_v(0) & \text{if } r = b_v \\ \infty & \text{otherwise} \end{cases} \tag{61}$$

Proof. From $V[v,0] = \{v\}$, $x_{vv} = 0$ and (52) it follows that $r = b_v$. Finally, the objective function in (44) amounts to $K_v(0)$. ■

5 Dynamic Programming Algorithm

Based on the relations which are derived in the preceding section, we are now able to formulate a dynamic programming algorithm. Recall that $g(v,i,s)$ is only defined for (v,i) with $v \in \mathcal{V}$, $0 \le i \le d_v$ and $0 \le s$ and $s + b_v + p_v \ll B$, whereas $h(v,i,r)$ is defined on (v,i) pairs with $v \neq 0$ and $b_v \le r \ll B$.

DYNAMIC PROGRAMMING ALGORITHM FOR ANIP

---

forall (v,i,s) with $v \in \mathcal{V}$, $0 \le i \le d_v$ and $0 \le s$ and $s + b_v + p_v \ll B$ **do**
  $g(v,i,s) = \infty$; /* *initialization* g */
forall (v,i,r) with $v \in \mathcal{V}$, $0 \le i \le d_v$ and $b_v \le r \ll B$ **do**
  $h(v,i,r) = \infty$; /* *initialization* h */
forall $v = n$ **downto** 0 **do begin**
  forall s with $0 \le s$ and $s + b_v + p_v \ll B$ **do**
    $g(v,0,s) = K_v(s + b_v + p_v)$;
  if $(v \neq 0)$ **then**
    $h(v,0,r_v) = K_v(0)$;
  forall $i = 1$ **to** d_v **do begin**
    forall s with $0 \le s$ and $s + b_v + p_v \ll B$ **do**
      $g(v,i,s) = \min\{g(s_v^i, d_{s_v^i}, 0) + g(v, i-1, s + p_{s_v^i}) + L_{s_v^i}(p_{s_v^i}),$
        $\min_\alpha\{h(s_v^i, d_{s_v^i}, \alpha) + g(v, i-1, s+\alpha) + L_{s_v^i}(\alpha)\}\}$;
    forall r with $b_v \le r \ll B$ **do**
      $h(v,i,r) = \min\{g(s_v^i, d_{s_v^i}, 0) + h(v, i-1, r - p_{s_v^i}) + L_{s_v^i}(p_{s_v^i}),$
        $\min_\alpha\{h(s_v^i, d_{s_v^i}, \alpha) + h(v, i-1, r-\alpha) + L_{s_v^i}(\alpha)\}\}$;
  end;
end;

When determining the coefficients $g(v,i,s)$ and $h(v,i,r)$ for $i > 0$, one needs to know all possible values of α which may occur. Weak bounds can be given, such as $\alpha \ge 0$, since obviously $\ell_{s_v^i}$ must be nonnegative, and $\alpha \ll B$, since all connections which pass the same edge are connected to the same concentrator, and hence, must fit the capacity of that concentrator. Of

course, the running time of the algorithm can be improved by exploiting better bounds. Since α must at least encompass the connections from demand node s_v^i, $\alpha \geq b_{s_v^i}$ would be a valid lowerbound. Furthermore, when determining $g(v,i,s)$, all connections on edge s_v^i must fit on the concentrator in node v. But the same holds for connections from vector s, the demand from node v itself, and the connections between the concentrator in v and its homing concentrator. Hence, $\alpha + s + b_v + p_v \ll B$ yields a valid upperbound on α for this case. Finally, when determining $h(v,i,r)$, the connections on edge s_v^i together with the demand from node v are only a part of the total number of connections r, hence, $\alpha + b_v \leq r$ imply a valid upperbound for α in this case.

In practical applications the number of different types of connections is usually small ($t = 2$). Furthermore if the applied cost structure is as indicated in Section 2, the general cost figures $K_w(k_w)$ and $L_e(\ell_e)$ can be determined easily (that is in small running time). The overall running time of the algorithm then equals $\mathcal{O}(nB^4)$, since the number of $T[v,i]$ trees we consider is $\mathcal{O}(n)$, for each tree we determine $\mathcal{O}(B^2)$ g-coefficients and $\mathcal{O}(B^2)$ h-coefficients. Moreover, each of these coefficients can be determined in $\mathcal{O}(B^2)$ time. The required storage space follows directly from the number of coeffiecients to be determined and is $\mathcal{O}(nB^2)$. This is formally stated in the following theorem, which is the main result of this paper.

Theorem 5.1 *Consider ANIP* as defined by (4)–(18), and with objective function defined by (20) and (21). If $t = 2$ (two types of connections and interface cards) and $\theta \leq |\mathcal{V}|$ (the number of concentrators bounded by the number of nodes in the tree), then ANIP* can be solved by a pseudopolynomial time dynamic programming algorithm which requires $\mathcal{O}(nB^4)$ time, and $\mathcal{O}(nB^2)$ storage space.*

When the algorithm is used as a tool in the planning phase of ATM network layout, there are several features which may be added to the algorithm. First, if one is interested in the optimal solution in which certain concentrators are already (pre)installed, one can easily incorporate this into the algorithm by adopting the cost function $K_w(k_w)$ for these specific nodes. A similar technique can of course be applied to prohibit the installation of concentrators in a node. Overall, this gives the possibility to incorporate partial solutions as part of the input. This can very well help to resolve many sensitivity questions which are extremely important in the planning of network structures.

Secondly, given a solution, it is easy to show the actual number of slots being used in a concentrator, as well as the number of Mb/s flowing over a set of connections. Hence, given a network strucuture, one can determine to what extend concentrators and connections are being used. This utilization level of the different components of the network may give insight into the stability of the network structure for future demand.

Thirdly, when one determines the optimal solution for the tree as a whole, but for different amounts of traffic from outside the tree (i.e. not only zero), this amount of traffic from outside the tree may be viewed as the number of connections incident to the root node which are used for routing in the Backbone. If the optimal solution is insensitive for this amount of traffic, this gives a justification for the approach to solve ATM network problems for the Backbone and trees separately.

problem	n	dp_aso_1 (s)	value
aso_1	12	0.37	92
aso_2	26	1.14	141
aso_3	29	1.71	161
aso_4	35	1.45	178
aso_5	28	0.96	151
aso_6	12	0.67	85
aso_7	22	1.51	147
aso_8	31	1.15	146

Table 1: Computational results for the instances from KPN (time measured in CPU sec.).

Finally, in practice the number of connections used is, although small (usually equal to two), of great importance for the running time of the algorithm, as it is proportional to the number of types available. By only allowing one type of connection (say the largest capacity), an approximation algorithm can be made which from a theoretical point of view may have a bad performance, but from a practical point of view can be very useful. By only considering one type of connection the running time decreases significantly, whereas the solutions obtained are feasible and usually capable of processing a somewhat larger demand than the demand which forms the input of the algorithm.

6 Computational Results

To test our algorithm on real-life instances from the Dutch telecommunication company, we implemented our algorithm in C++ on a DEC 2100 A500MP workstation with 128Mb of internal memory. All problem instances we consider have a cost structure as the one proposed in Section 2 and Section 3. Table 1 reports on eight problem instances, for which three different concentrators were available, with capacities 3,10 and 20 slots, respectively.

Table 2 reports on the same instances but with the largest concentrator capacity being 50 instead of 20 (B increases). The results show that B is indeed an important determinant for the running time of the algorithm. The optimal solutions have not changed, which only indicates that capacity on the concentrators was not a restraining factor.

From the above results it follows that if extremely large trees are considered, together with even larger capacity concentrators, the running times of the algorithm might become unacceptable. Therefore we have also implemented an algorithm *dp_aso_2* in which only one type of connection (155Mb/s connections) is considered (as explained in Section 5). This relaxation causes the number of g and h-coefficients which have to be determined to be reduced significantly. Table 3 shows that the running times of the algorithm behave accordingly. More importantly, although the algorithm does no longer need to provide an optimal solution, the solutions found by the algorithm proved to be very similar to the optimal solutions and still usefull in practice.

problem	n	dp_aso_1 (s)	value
aso_1	12	11.59	92
aso_2	26	29.57	141
aso_3	29	50.59	161
aso_4	35	37.83	178
aso_5	28	25.14	151
aso_6	12	18.53	85
aso_7	22	42.51	147
aso_8	31	31.68	146

Table 2: Computational results for the instances from KPN with larger capacity concentrator.

problem	n	dp_aso_2 (s)
aso_1	12	0.02
aso_2	26	0.03
aso_3	29	0.03
aso_4	35	0.02
aso_5	28	0.02
aso_6	12	0.02
aso_7	22	0.02
aso_8	31	0.03

Table 3: Computational results for the instances from KPN with only one type of connections.

problem	n	dp_aso_1 (s)
aso_17	34	9.05
aso_24	61	12.90
aso_58	59	10.74
aso_68	43	8.79

Table 4: Computational results for the instances from KPN with larger capacity concentrator.

Finally, some larger trees were considered. Table 4 states the results for these larger instances, where again three concentrator capacities were considered, with capacities 3, 10, 30, respectively.

Concludingly, the algorithmic ideas presented in this paper prove to be very useful in practice. The algorithm yields optimal solutions for real-life problem instances, which are of significant problem size. Moreover, additional features can be added to the planning tool if desired, making the tool even more attractive as a decision support system in the network design planning phase.

References

[1] A. Balakrishnan, T.L. Magnanti, A. Shulman, and R.T. Wong. "Models for planning capacity expansion in local access telecommunication networks". *Annals of Operations Research*, 33:239–284, 1991.

[2] A. Balakrishnan, T.L. Magnanti, and R.T. Wong. "A decomposition algorithm for local access telecommunications network expansion planning". *Operations Research*, 43(1):58–76, 1995.

[3] D. Bienstock, S. Chopra, O. Günlük, and C.-Y. Tsai. "Minimum cost capacity installation for multicommodity network flows". Working paper, July 1995.

[4] D. Bienstock and O. Günlük. "Capacitated network design — polyhedral structure and computation". Working paper, June 1995.

[5] B. Brockmüller, O. Günlük, and L. Wolsey. "Designing private line networks - polyhedral analysis and computation". Discussion Paper 9647, Center for Operations Research and Econometrics, October 1996.

[6] G. Cho and D.X. Shaw. "Limited column generation for local access telecommunication network design - formulations, algorithms, and implementation". Working Paper, January 1995.

[7] O. Flippo, A. Kolen, A. Koster, and R. van de Leensel. "A dynamic programming algorithm for the local access network expansion problem". Research Memorandum 96/027, Maastricht University, 1996.

[8] M. R. Garey and D.S. Johnson. *"Computers and intractability: a guide to the Theory of NP-Completeness"*. Freeman and Company, N.Y., 1979.

[9] B. Gavish. "Topological design of telecommunication networks: Local access design networks". *Annals of Operations Research*, 33:17–71, 1991.

[10] B. Gavish. "Topological design of computer communication networks - the overall design problem". *European Journal of Operational Research*, 58(2):149–172, 1992.

[11] B. Gavish and K. Altinkemer. "Backbone network design tools with economic tradeoffs". *ORSA Journal on Computing*, 2(3):236–252, 1990.

[12] D.S. Johnson and K.A. Niemi. "On knapsacks, partitions and a new dynamic programming technique for trees". *Mathematics of Operations Research*, 8:1–14, 1983.

[13] T.L. Magnanti, P. Mirchandani, and R. Vachani. "The convex hull of two core capacitated network design problems". *Mathematical Programming*, 60:223–250, 1993.

[14] T.L. Magnanti, P. Mirchandani, and R. Vachani. "Modelling and solving the two-facility capacitated network loading problem". *Operations Research*, 43:142–157, 1995.